MEASURING INTERFACIAL TENSION WITH THE PENDANT DROP METHOD

A Thesis
Presented to
The Academic Faculty

by

Kevin K. Mohan-Nair

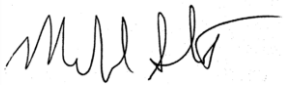
In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science in Electrical Engineering in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2014

MEASURING INTERFACIAL TENSION WITH THE PENDANT DROP METHOD

Approved by:

Dr. Alberto Fernandez-Nieves, Advisor
School of Physics
Georgia Institute of Technology



Dr. Michael Schatz
School of Physics
Georgia Institute of Technology

Date Approved: December 12, 2014

ACKNOWLEDGEMENTS

First and foremost, this work is dedicated to my Mother, Dr. Maya N. Nair, Grandfather, Dr. K Narayanan Nair, and Grandmother, Ramani Nair. Each of them has helped me unequivocally on my journey and I truly could not have done this without their help. I'd like to thank my thesis advisor, Dr. Alberto Fernandez-Nieves, for all of his encouraging guidance and support throughout the period of research. I would also like to thank Dr. Michael Schatz for his time helping me write this thesis and reviewing my work. A special thanks to two distinguished postdoctoral fellows, Dr. Ekapop Pairam and Dr. Josefa Guerrero, who have helped me immensely with the theory and coding the solution. Finally, I would like to thank everyone in the Georgia Tech Soft Condensed Matter Lab, the Georgia Tech School of Physics, as well as the Georgia Institute of Technology for aiding me in the process.

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

β	Shape Parameter, also known as the Lorentz Factor
γ	Interfacial Tension
R_0	Radius of curvature at the drop apex
ρ	Fluid Density
D_s	Drop Aspect Ratio upper distance
D_e	Drop Aspect Ratio lower distance
IFT	Interfacial Tension

SUMMARY

In the field of soft condensed matter and in particular, microfluidics, the understanding of surface tension is vital. Interfacial tension (IFT), fundamentally defines liquid interactions at the micro-scale. To give some perspective of the importance of understanding this tension, gravity hardly plays a role given the tiny masses of micro particles when compared to the role that surface tension plays. Most of the current studies on IFT rely on some application of the differential Young-Laplace equation, which expresses the equilibrium condition across an interface. The application of the Young-Laplace equation goes as far as to state the fundamental set of differential equations that define the forces and geometries of the traditional pendant drop, however, it uses an approximate shape parameter that is taken from empirical data. Using the power of modern computing power, the research will analyze the second order partial differential Young-Laplace Equation in its association with the interfacial tension between different interfaces using a computer program written in MATLAB. The differential Laplace Equation will be applied in conjunction with image processing and other data processing methods to develop a real time user-friendly IFT calculator that takes in an image of a liquid immersed in another liquid and outputs a value of the interfacial tension between the two immiscible fluids all in real time.

CHAPTER 1

INTRODUCTION

In the field of soft condensed matter and in particular, microfluidics, the understanding of surface tension is vital. Interfacial tension fundamentally defines liquid-liquid interactions at the micro-scale. Over the last decade, computers have continued following a Moore's Law trend and have increased in computational power tremendously. Following this increase in computational power was the subsequent boom in the levels of complexity of various computer software. Specifically, programs like MATLAB have enabled even the average user to model, design, and analyze high quality renderings of data with industry standard precision and accuracy.

Current research in the field focuses on a variety of different methods to measure interfacial tension. One such method is the spinning drop method in which a drop of a fluid is dispensed in a chamber that contains a rod of known surface area which begins to spin at an increasing rate until the fluid that has wetted the rod begins to detach. This centripetal force is equated to a surface tension of the fluid using the rotation rate of the rod and the surface area of the rod. Another method uses a variation of the pendant drop method called the sessile drop method, which is very similar to the pendant drop method, except for the fact that the drop is at rest on a surface instead of hanging from a medium. Several of the variables in calculating IFT using the sessile drop method are shared with those outlined in the pendant drop method such as the radius of curvature, R, of the pendant drop at the apex of the drop, and the form factor constant, β, which defines the shape of all possible shapes of drops of any fluids under the presence of gravity [1]. Most of the current studies on IFT rely on some form of the differential Young-Laplace

equation (show below), which is what fundamentally drives this research. There have been a few studies that have attempted applying similar image processing techniques with the sessile drop method and other studies that have only approximated the IFT calculations [2].

$$\gamma = \frac{\Delta \rho g R^2}{\beta} \qquad \text{(Eq. 1)}$$

Classically, proposed methods to analyze the interfacial tension between two interfaces have relied on approximative methods and iterative numerical computations with limited understanding of the physics that fundamentally defines these interactions. The Young-Laplace equation goes as far as to state the fundamental set of differential equations that define the forces and geometries of the traditional pendant drop, however, it uses an approximative shape parameter that is taken from empirical data [3]. The pendant drop method for measuring interfacial tension with computational processing will ideally allow for a better calculation of IFT rather than the approximation method using empirical data. It is in this marriage between modern computational power and understanding of fundamental soft condensed matter physics that the pendant drop method for measuring interfacial tension arises.

Using the power of modern computing power, the research will analyze the second order partial differential Young-Laplace Equation in its association with the interfacial tension between different interfaces using a computer program written in MATLAB. The differential Laplace Equation will be applied in conjunction with image processing and other data processing methods to develop a real time user-friendly IFT calculator that takes in an image of a liquid immersed in another liquid and outputs a

value of the interfacial tension between the two immiscible fluids all in real time. The impact of such a real time calculator for interfacial tension has many potential applications in diverse industries ranging from food science to chemicals manufacturing.

CHAPTER 2

LITERATURE REVIEW

Calculating interfacial tensions (IFTs) for pendant drop systems has been studied and accomplished to relatively high degrees of accuracy in the past; however, many of the methods used in those studies primarily used various geometric properties of a drop to approximate an accurate value for IFT. There has not been a great deal of work done with form fitting the profile of a drop to get precise IFT values.

One of the research studies talks about the use of axisymmetric drop analysis and proposes a unique way of processing the drop shape [2]. This method is outlined in the paper by Del Rio and talks about the development of a new method to gauge pendant drop topologies for interfacial tension measurements. The method is formally called axisymmetric drop shape analysis (ASDA) and it fits the Laplace equation of capillarity to an image of a pendant drop and results in a value for interfacial tension. The second method used in this paper is called ASDA-HD, where the H and D stand for height and diameter, and increases the accuracy of the regular ASDA method by using two more parameters to determine a shape parameter.

The strengths of this method is the increased accuracy in measuring the interfacial tension of a pendant drop or a sessile drop using a more advanced axisymmetric model than what the traditional methods call for. One of the weaknesses of this paper is the fact that the study uses two extra parameters (height and diameter) to try to characterize the shape parameter used to calculate an interfacial tension [2]. This could be improved upon since the height and diameter do not characterize a pendant drop's full shape and is approximate to that extent. The treatment of the diameter and height parameters will be

helpful to the overall context of characterizing the pendant drop. Furthermore, this paper takes an in depth analysis into the boundary conditions problem at multiple points, which is useful to the context of the study overall.

A second paper by Saad talks about the shape parameter used in surface tension and interfacial tension measurements [4]. The paper compares and contrasts several of the traditional as well as several of the recent advances in ways to characterize the shape parameter using the geometry of the pendant drop. Furthermore, the paper talks about the accuracy and shortcomings of most shape parameter methods when the pendant drop approaches an ideal spherical shape. Finally, the paper makes an important note stating that the ideal conditions for measuring a shape parameter for a drop are when the drop holder has a flat, horizontal, and circular surface with sharp edges.

This paper's primary strength is in its use of analysis techniques to test several important methods of shape parameter calculation for pendant and sessile drops [4]. Another strength of the paper is that it gives several tables and graphs that show results from studies on shape parameter as a function of bond number and an assortment of other useful parameters. One of the weaknesses of this paper, however, is that it doesn't provide much of a new method for solving for a shape parameter; rather, this paper simply compares and contrasts other studies without suggesting new techniques.

The paper by Peters about finding an IFT with a modified contour method talks about the use of a pressure measurement in conjunction with images of pendant drops in the oil-water interface to come up with a value of the interfacial tension [5]. This method is quite novel because it can come up with a value for interfacial tension without having to apply the Laplace equation. The paper uses a combination of image processing and the study of

three dimensional shapes to measure the volume difference and pressure difference of the different interfaces to find an interfacial tension.

A strong point about this paper is that it talks about the use of a system that does not involve the primary form of the Laplace equation, which is used in most other surface tension measurements. The presentation of the data is well laid out and it is easy to follow to see how they computed an interfacial tension without using the Laplace equation. One of the weaker points of this paper was that it had to use iterative methods to find the value of interfacial tension rather than compute one directly given all the parameters to the system. Overall, this paper presents an entirely non-traditional iterative method to find the interfacial tension and gives yet another set of numbers to compare results and to provide another measure of the accuracy.

The study on a finite element based algorithm to determine IFT by Dingle et al. talks about the Galerkin finite element method to solve the axisymmetric form of the Young-Laplace equation [1]. It revolves around the idea that one can estimate the shape parameter by minimizing the difference between the theoretical and experimental shape functions via the three arc length based first order ordinary differential equation (ODE) solution. In addition to the application of the Bashforth and Adams (BA) algorithm, this study goes to propose a more robust method using boundary conditions and a second order arc length solution.

The first of many strengths of this study is the application of the solution to the three arc length second order ODE with boundary conditions at the drop apex and at the contact line of the drop to the nozzle [1]. This is not a trivial practice and the solution to the three arc length second order ODE gives important data on the drop geometry. A

second strong point is that the solution is robust because it can be applied to drops of almost any material of almost any viscosity, which is important for developing an application that is meant to be versatile and applicable to any pendant drop system. Finally, the method can tell whether the drop shape is in static equilibrium or if it is still in the process of morphing into its ultimate equilibrium shape. One of the weaker attributes of this paper, on the other hand, is that the more total elements ran, the longer the program used the computer's resources to process a solution. Furthermore, the two parameter fit was the least precise; however with increasing parameters, the values for shape parameter were in agreement with those found in literature to less than 0.5% error.

The fact that the study uses another method to solve the Young-Laplace equation gives another avenue to cross check results. Secondly, the study uses the water-air interface, which is a pendant drop system that has been well studied by almost every other paper in literature relating to the pendant drop. Finally, the study uses a minimization of error step in which the shape parameter obtained from experimental data collection is compared to the theoretical solution of the shape parameter from the Young-Laplace equation and is optimized using nonlinear least squares regression.

In the paper by Alvarez, low Bond number drop shapes are analyzed and the study talks about the use of a non-gradient based algorithm to compute the interfacial tension between a liquid-liquid or gas-liquid interface [3]. The traditional methods for calculating the interfacial tension using Bond numbers, also known as the shape parameter, break down when the Bond number tends towards tiny numbers near zero. The paper investigates a new method based on the Nelder-Mead simplex method to solve the least squares problem. Their method generates values within 0.1% of the literature

values for interfacial tensions of well-known liquid-liquid and liquid-air systems. One of the strong points about this paper is the presentation of the data and the graphical abstract that it provides. This paper also discusses the ability to calculate interfacial tensions in systems of liquids with similar density and also in cases where the drop shape is approximately spherical [3]. This is something that virtually no traditional solution to the Young-Laplace equation could achieve. One of the weaker points of this paper is the fact that to compute the interfacial tension to high precision for nearly spherical drops, the computer program had to run for a much longer time with many more data points to utilize. There have been cases where traditional methods to calculate an interfacial tension do not work due to the nearly spherical drop shape resulting in an almost zero bond number, and this paper has shed some light on how to solve for an interfacial tension for such systems.

CHAPTER 3

EXPERIMENTAL SETUP

The research project will be divided into four major categories. The first category will be image acquisition. This was accomplished by taking a CCD camera mounted on a rigid steel frame kept a fixed distance from a container holding the pendant drop. The camera will be focused at the tip of the syringe where the pendant drop will be formed for best drop profile capture. In addition to a fixed position imaging system, to keep consistency between data acquisition steps, there will be a light diffuser stationed behind the drop to diffuse the light from the light source illuminating the drop from behind. Furthermore, a fluid pump will be used to pump the inner fluid of the pendant drop system at a constant flow rate in order to provide the ability to study the interfacial tension of a pendant drop system as a function of time. The experimental set up described is shown in Figure 2:

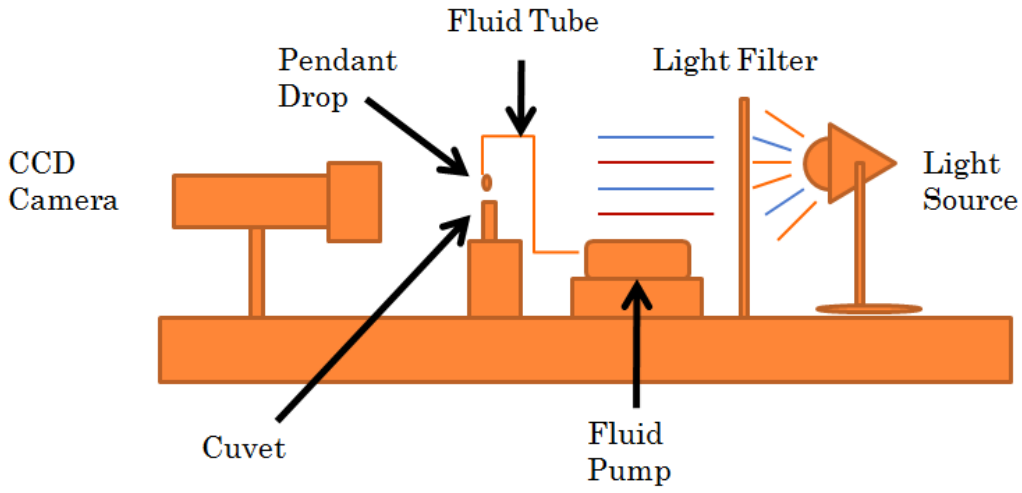


Figure 3.1 – Experimental Setup Diagram

The second part of the experiment is the image processing step in which the data acquired through image capture described above is manipulated to provide a value for interfacial tension of a particular pendant drop system. The program running the image processing should ideally provide a theoretical value of interfacial tension for a particular system alongside the calculated value using the image processing techniques.

The third phase of the experiment will be to compare several images and the resulting interfacial tension values with known literature values of IFT to test the accuracy and precision of the program. This validity check will hopefully point out any errors or will highlight fundamental limitations of the program in its ability to calculate IFT values.

The fourth and final phase of this project will involve compiling all data obtained and measured into a simple user-friendly graphical user interface (GUI). Preferably, the MATLAB GUI tool will be used in order to export the project as an executable windows file for use on multiple computer systems.

The materials used will be a CCD camera and its computer software, a light source, a light diffuser, a fluid pump, fluid tubing, syringe with 0.7mm diameter tip, a 4020 aluminum frame to hold each component of the setup, and inner/outer fluids, as outlined in Figure 2. The inner and outer fluids that will primarily be used are air, water, ethanol, PDMS silicon oil of various cPs ranging from 10 cPs to 1000 cPs, and finally, carbopol.

CHAPTER 4

DESIGN & IMPLEMENTATION IN MATLAB

To solve the Young-Laplace equation for interfacial tension (γ), the density difference of the two fluids (Δρ), the radius of the curvature of the circle of best fit at the apex of the drop (R_0), and the shape factor (β) must be known. These parameters will give the interfacial tension between two fluids as seen from Equation 1. Another good method to cross-compare interfacial tension is shown in Equation 2 below, which shows the DS/DE aspect ratio method which uses two parts of the drop and the empirical equation to calculate the shape parameter, β [6].

$$\beta = 0.12836 - 0.7577\frac{D_S}{D_E} + 1.7713\left(\frac{D_S}{D_E}\right)^2 - 0.5426\left(\frac{D_S}{D_E}\right)^3 \qquad \text{(Eq. 2)}$$

The equation relates the drop height and the drop width taken at the apex of the drop. As shown in the following figure.

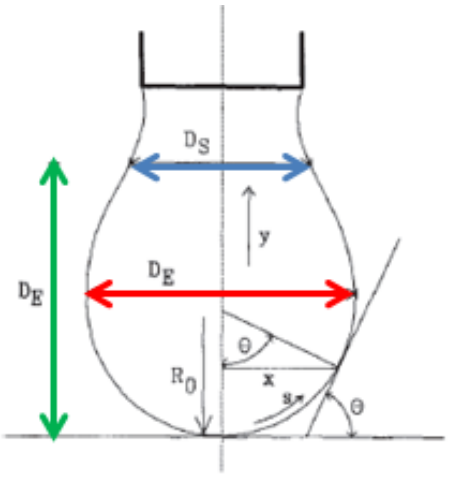


Figure 4.1 – DS/DE aspect ratio.

This approximation method for the shape parameter is quite accurate in giving a correct value; however the explicit values of the Lorentz Factor are given by the three equations below.

$$dx/ds = \cos\theta \quad \text{(Eq. 3)}$$

$$dz/ds = \sin\varphi \quad \text{(Eq. 4)}$$

$$d\varphi/ds = 2 + \beta z - \sin\varphi/x \quad \text{(Eq. 5)}$$

These equations dictate the behavior of the pendant drop system and are the fundamental equations from which Equation 1 comes from. As shown in the figure below, we have the different β curves plotted at intervals chosen to show discreteness. The most inwards numbers of the shape parameter are smaller and those that tend to the outside are close to a shape factor of one. Since the equation of interfacial tension depends on this, according to equation 1, a larger value of the shape factor will scale down the interfacial tension of the system while a smaller value of beta can drastically increase the value of the interfacial tension.

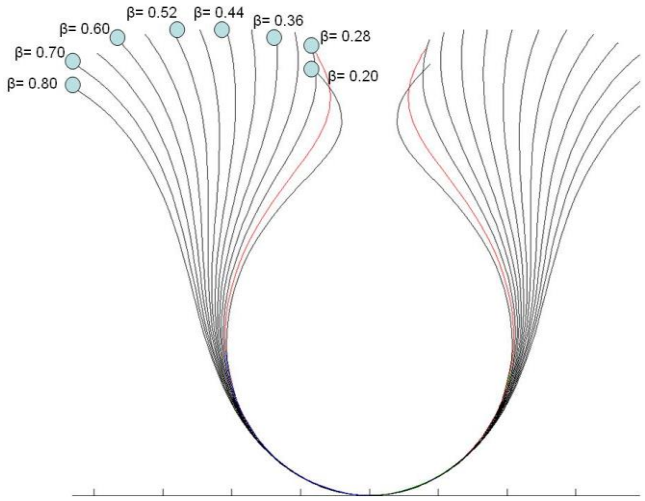


Figure 4.2 –Plot of different Beta curves

The DS/DE method for approximating the shape factor is a good comparison to track the validity of the experimental methods used in the Interfacial Tension Calculator. For this reason, it's important to present this calculation in the program run. The program "Interfacial Tension Calculator" would begin by accepting these inputs as shown in Figure 4.3.

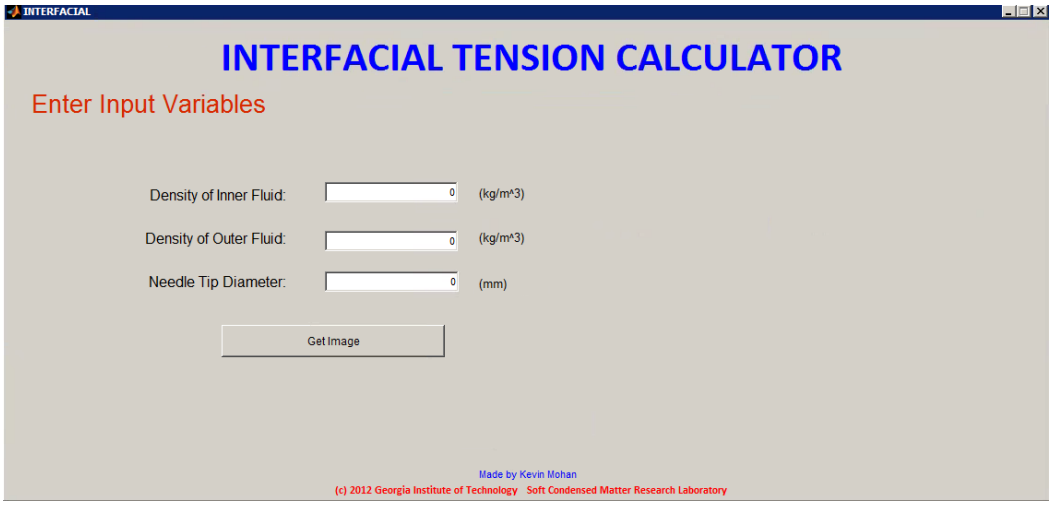


Figure 4.3 - Interfacial Tension Calculator startup page.

Next, the image must be inputted into the program for processing, shown below:

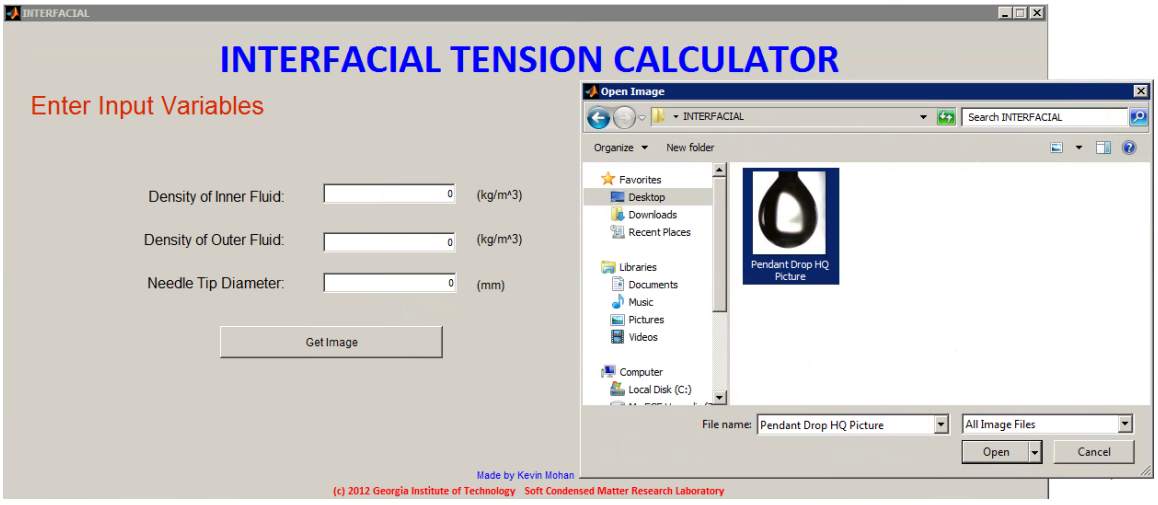


Figure 4.4 – Selecting "Get Image" brings up a menu to select an image for processing.

The following figure show the final input to the program along with an image that previews the image to run in the program.

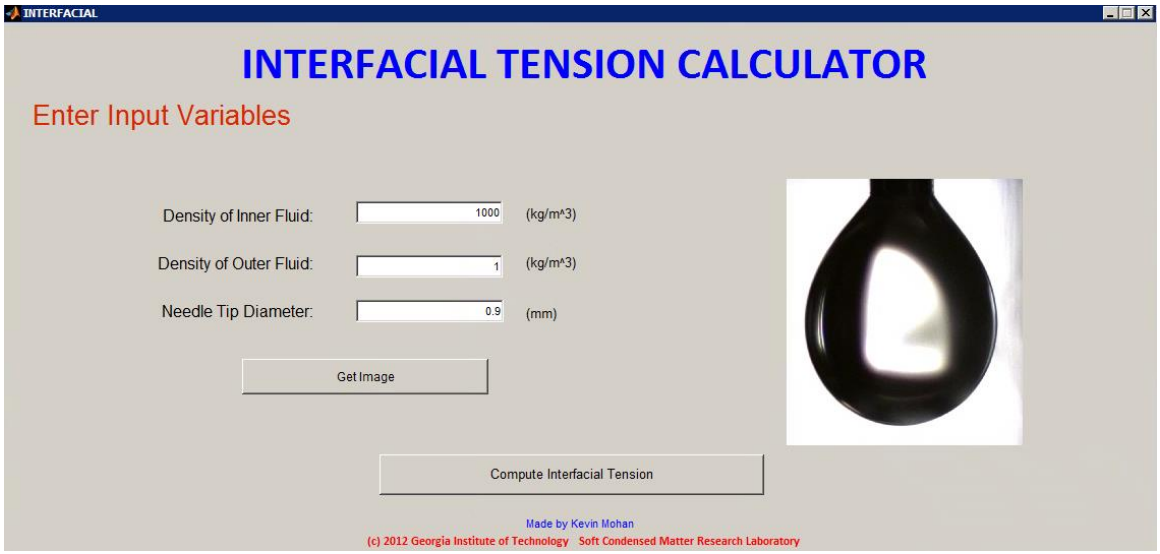


Figure 4.5 – Four inputs to the program have been entered and are ready to compute.

Next, the program is run and the following figure is generated. It shows the contour of the fluid system at the interface. This contour is overlaid on top of the image for better visibility and a visual check on the fit of the curve.

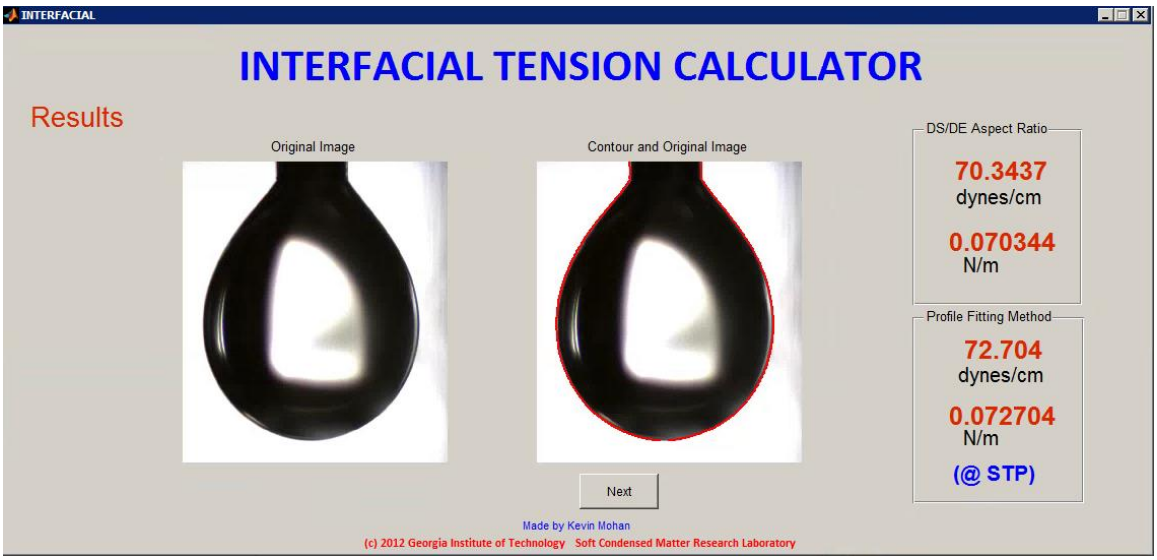


Figure 4.6 – Original image with edge-detected contour overlay and IFT final result.

Next, we plot the radius of curvature of best fit inside the drop contour to show the value that worked best according to a least squares minimization. Figure 4.7 shows this plot as well as the zoomed in region of the bottom of the drop for better overall visual confirmation of the goodness of fit.

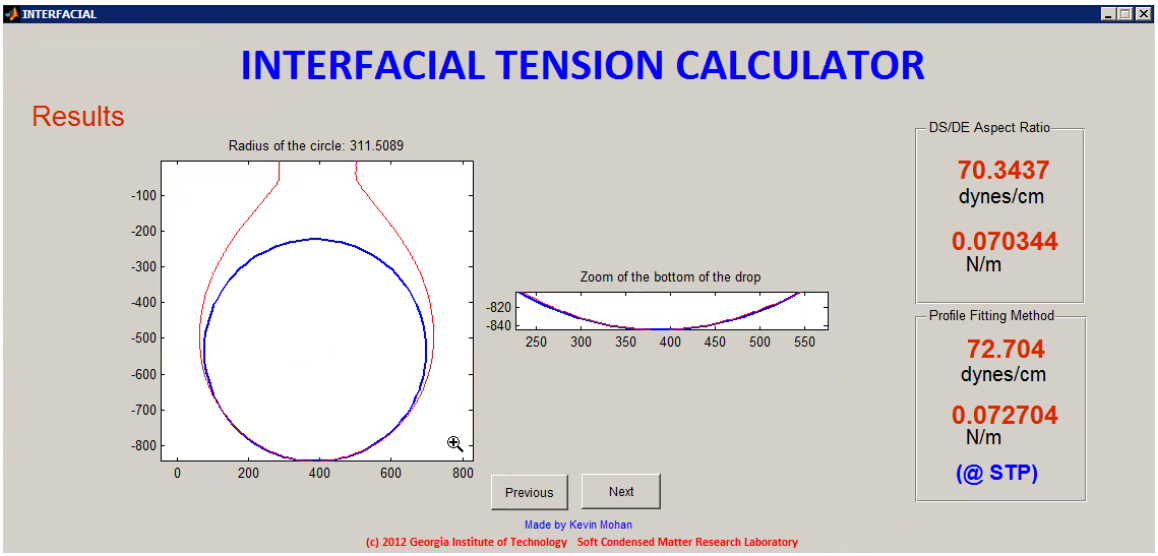


Figure 4.7 – Plot of radius of curvature of best fit at the drop apex with magnified region.

As mentioned previously, the DS/DE method [6] of gauging the value of the shape factor is helpful as a comparison value to both the experimentally generated values from running the program and the literature values that have been confirmed by high-tech facilities. Figure 4.8 shows this comparison as well as the least squares minimization of the beta value. The top right values in red show the DS/DE method values for IFT and the lower values are those that are found using the profile fitting method found in the Interfacial Tension Calculator.

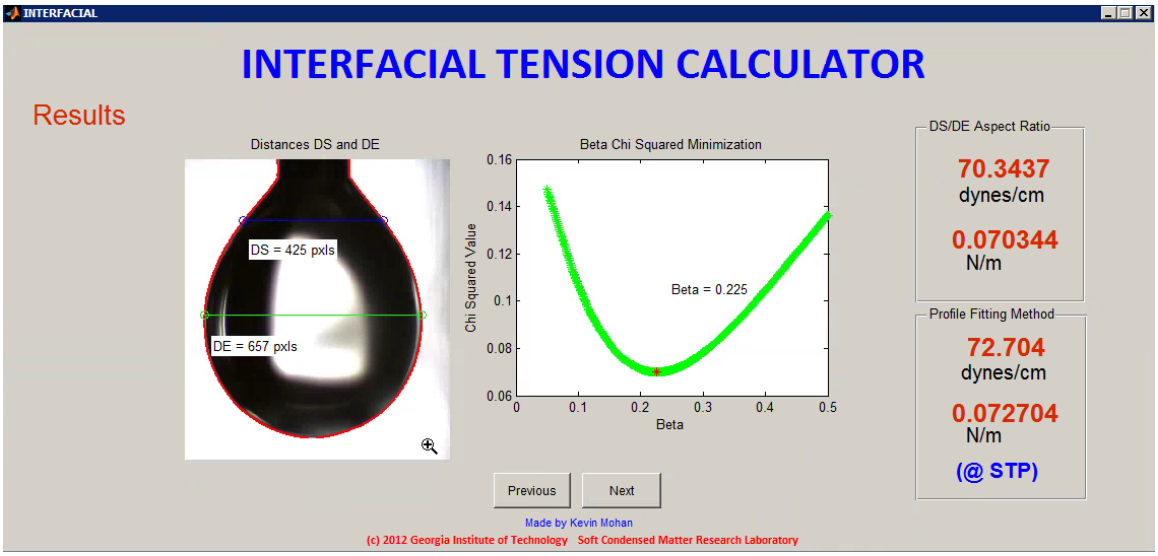


Figure 4.8 –DS/DE aspect ratio calculation and Beta Chi Square Minimization.

Finally, the program shows the overall smoothed contour along with the initially fitted contour to gauge the goodness of the run of the program. This is shown below in Figure 4.9.

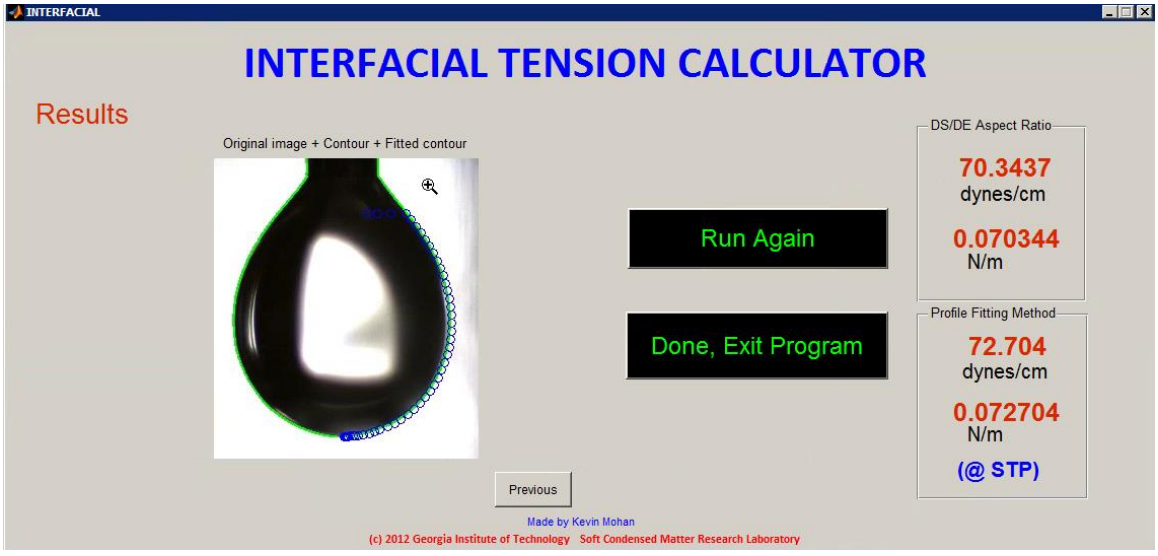


Figure 4.9 – Original Image with fitted contour along with a "Run Again" option.

CHAPTER 5

DISCUSSION AND CONCLUSION

After running several different fluid-fluid systems in the experimental setup, several hundred images had been captured and several images had been chosen to process using the Interfacial Tension Calculator. In almost every case, the calculator ran without error and the result that was computed for the interfacial tension using the experimental method which I developed was within 3-5% of the actual literature value of that particular fluid-fluid system. In some cases, the experimental method used in the interfacial tension calculator program was even closer to the literature values than the DS/DE aspect ratio calculation.

For images that did not fit the standard drop shape given by the beta parameter curves, the program would crash or would give an unprecedented value of interfacial tension. At times, the value of the interfacial tension would be higher or much lower than the expected values and this was primarily due to the fact that there are various curve fitting algorithms in the code that could have potentially altered values elsewhere which lead to the high or low IFT values. While occasional, there were times that the code would give much higher and lower values due to the images themselves either being of low quality (resolution and magnification) or if the materials that were photographed had high viscosity or were difficult to handle because they would wet the surface of the needle tip and cause an uneven drop picture.

Overall, the Interfacial Tension Calculator runs as intended and is good for a first delve into finding the interfacial tension of a fluid-fluid system. In the future, this system could be developed further, leading to more accurate and precise measurements.

APPENDIX A

MATLAB CODE

INTERFACIAL.m

```
%Created by Kevin Mohan of the GT Soft Condensed Matter Lab
%Last modified on: 12/01/2014
function varargout = INTERFACIAL(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @INTERFACIAL_OpeningFcn, ...
                   'gui_OutputFcn',  @INTERFACIAL_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function INTERFACIAL_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
initialize_gui(hObject, handles, false);
function varargout = INTERFACIAL_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
clc;clear secondarycalc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function density1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function density1_Callback(hObject, eventdata, handles)
global density1
density1 = str2double(get(hObject, 'String'));
if isnan(density1)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
handles.metricdata.density1 = density1;
guidata(hObject,handles)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
function density2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function density2_Callback(hObject, eventdata, handles)
global density2
density2 = str2double(get(hObject, 'String'));
if isnan(density2)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
handles.metricdata.density2 = density2;
guidata(hObject,handles)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
function needleTipDiameter_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function needleTipDiameter_Callback(hObject, eventdata, handles)
global needleTipDiameter
needleTipDiameter = str2double(get(hObject, 'String'));
if isnan(needleTipDiameter)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
handles.metricdata.needleTipDiameter = needleTipDiameter;
```

```
guidata(hObject,handles)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
function calculate_Callback(hObject, eventdata, handles)
set(handles.uipanel1,'Title','Processing Image...');
set(handles.origimage,'Visible','off');
set(handles.calculate,'Visible','off');
set(handles.density1,'Visible','off');
set(handles.density2,'Visible','off');
set(handles.getImage,'Visible','off');
set(handles.needleTipDiameter,'Visible','off');
set(handles.text5,'Visible','off');
set(handles.text17,'Visible','off');
set(handles.text18,'Visible','off');
set(handles.text16,'Visible','off');
set(handles.text1,'Visible','off');
set(handles.text2,'Visible','off');
global needleTipDiameter
global density1
global density2
global inputImgOriginal
global adjustprofilex
global betaminind
global rad
global xcrop
global adjustprofiley
global ycrop
global ytemp
global xcoordr
global ycoordr
global a0
global mx
global xcoordl
global ycoordl
global deind
global deVert
global de
global ds
global bi
global stddev
global betaval
inputImg = bwperim(~edge(uint8(imfill(~im2bw(inputImgOriginal),'holes'))));
[col,row]=size(inputImg);
img = inputImg(2:col-1,2:row-1); %for some reason, we get a white outline on image, so we trim the
image
[col,row]=size(img);
leftOutlineColumn = []; rightOutlineColumn = []; DBLR = [];
for i=1:col
    for j=1:row
        if img(i,j) == 1
            leftOutlineColumn = [leftOutlineColumn j]; break
        end
    end
    for k=row:-1:1
        if img(i,k) == 1
            rightOutlineColumn = [rightOutlineColumn k]; break
        end
    end
    DBLR = [DBLR abs(j-k)];
    if DBLR(i) == (row-1)
        DBLR(i) = 0;
    end
end
ratio1 = DBLR(1); %% Needle diameter. We use this number to the conversion from pixel to m

% Now we find the contour

[numrows,numcols] = size(img);
for rowindex = 1:numrows
    for columnindex = 1:numcols
        if(img(rowindex,columnindex) == 1)
            xcoordr(rowindex) = columnindex;
            ycoordr(rowindex) = rowindex;
        end
    end
end
for rowindex = 1:numrows
    for columnindex = numcols:-1:1
        if(img(rowindex,columnindex) == 1)
            xcoordl(rowindex) = columnindex;
            ycoordl(rowindex) = rowindex;
```

```
        end
    end
end

%Plots: picture and picture+contour

%     figure;
%     imshow(inputImgOriginal);title('Original Image');hold off;
%     figure;
%     imshow(inputImgOriginal);title('Contour and Original Image')
%     hold on;plot(xcoordr,ycoordr,'r','LineWidth',2);hold
on;plot(xcoordl,ycoordl,'r','LineWidth',2);hold off;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
% Radius of the circle that fits with the drop

totalradxtest = [];

for i = round(2*length(xcoordr)/3):length(xcoordr)-1
    x1temp = xcoordr(i:length(xcoordr))';
    y1temp = ycoordr(i:length(xcoordr))';
    x2temp= xcoordl(i:length(xcoordr))';
    y2temp = ycoordl(i:length(xcoordr))';

    % We smooth this data and do subpixel interpolation
    xtemp=[x1temp;x2temp];
    ytemp=[y1temp;y2temp];

%     figure
%     plot(xtemp,ytemp,'.')
%     hold off

    [xtemp,m,n]=unique(xtemp);
    ytemp=ytemp(m);
    f2 = fit(xtemp,ytemp,'spline');
%     figure
%     plot(f2,xtemp,ytemp)
%     hold on
%   plot(xtemp,ytemp,'o')
%     hold off
    x=xtemp(1):0.1:xtemp(end);
    y=f2(x);
    x=x';

%     figure
%     plot(x,y,'go')

    mx = mean(x);
    my = mean(y);
    X = x - mx;
    Y = y - my;
    dx2 = mean(X.^2);
    dy2 = mean(Y.^2);
    t = [X,Y]\(X.^2-dx2+Y.^2-dy2)/2;
    a0 = t(1);
    b0 = t(2);
    totalradxtest(i)=sqrt(dx2+dy2+a0^2+b0^2);
    npoints(i)=2*(length(xcoordr)-1-i);
    clc;
end

% Curvature vs number the points used to fit
elm=find(totalradxtest>0);
npoints=npoints(elm);
radxtest=totalradxtest(elm);
% figure
% plot(npoints,radxtest,'.')
% title('Curvature')
% hold off

% The have to find the plateau of the former plot to get the real value of
% the radius

n=10;% we take segments which length is n pixels
for i = 1:length(npoints)-n
```

```
    slope(i) = (radxtest(i+n)-radxtest(i))/(n);
    if abs(slope(i))< 0.05
        fslop(i)=slope(i);
        xslop(i)=npoints(i);
        curv(i)=radxtest(i);
    end
end
rad=mean(radxtest); %% Radius of the circle that fit with the drop

%     figure;
% %     circle = rsmak('circle',rad, [a0+mx b0+my-44]);
%     circle = rsmak('circle',rad, [a0+mx -max(ytemp)+rad]);
%     circ2 = fncmb(circle);
%     fnplt(circ2);hold on;axis equal;
%     plot(xcoordr,-ycoordr,'r');hold on; plot(xcoordl,-ycoordl,'r');
%     title(['Radius of the circle: ',num2str(rad)]);hold off
%
%     figure;
%     circle = rsmak('circle',rad, [a0+mx -max(ytemp)+rad]);
%     circ2 = fncmb(circle);
%     fnplt(circ2);hold on;axis equal;
%     plot(xcoordr,-ycoordr,'r');hold on; plot(xcoordl,-ycoordl,'r');
%     axis([max(xcoordl)*0.6 max(xcoordr)*0.8  -max(ycoordr) -0.95*max(ycoordr)]);
%     title('Zoom of the bottom of the drop');hold off


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
% First method to obtain beta: from the distances DS and DE

    for i=1:length(xcoordr)
        xdif(i)=abs(xcoordr(i)-xcoordl(i));
    end

[de,deind]=max(xdif);
deVert=length(ycoordr)-de;
ds=xdif(deVert);

beta = 0.12836 - 0.7577*(ds/de) + 1.7713*(ds/de)^2 - 0.5426*(ds/de)^3;
% figure;
% imshow(inputImgOriginal);title('DS DE')
% hold on;plot(xcoordr,ycoordr,'r','LineWidth',2);hold on;plot(xcoordl,ycoordl,'r','LineWidth',2);hold
on;
% hold on; plot([xcoordl(deind) xcoordr(deind)],[ycoordl(deind) ycoordr(deind)],'go-');hold on;
% plot([xcoordl(deVert) xcoordr(deVert)],[deVert deVert],'bo-');
% title('Distances DS and DE');text(90,de-90,['DE = ',num2str(de)],'BackgroundColor',[1 1 1]);
% text(deVert + 15,ds-150,['DS = ',num2str(ds)],'BackgroundColor',[1 1 1]);
% hold off;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
% Second method to obtain beta: solve Young-Laplace equation

global b123;
counter = 1; Yall = [];
for b123 = 0.05:0.001:0.5
    [T Y] = ode45(@rigid, [0 3.1415926535897932], [10^-100,10^-100,10^-100]);
    Yall(:,:,counter) = Y;
    counter = counter + 1;
end
for temp = 1:counter-1
    xbeta(:,temp) = Yall(:,2,temp);
    ybeta(:,temp) = Yall(:,3,temp);
    %   plot(Yall(:,2,temp),Yall(:,3,temp),'g');axis equal;hold
    %   on;plot(-Yall(:,2,temp),Yall(:,3,temp),'g');       % Take care if you
    %   uncomment this plot because it's inside a loop
end

% Crop the top part of the drop to eliminate the needle part
ycrop = ycoordr(length(ycoordr)-round(0.8*length(ycoordr)):length(ycoordr));
xcrop = xcoordr(length(xcoordr)-round(0.8*length(xcoordr)):length(xcoordr));

% Make the contour nondimensional, and traslading the apex to (0,0)
xnorn=(xcrop-min(xcrop))/rad;
ynorn=(ycrop-min(ycrop))/rad;
ynorn = ynorn(end:-1:1); %% to get the bottom of the drop at the bottom of the image
%     figure
```

```
%     plot(xnorn,ynorn,'-r')

% To compare the contour obtained from the image with the calculated one,
% we need to have the same number of points. In our case, the experimental
% contour has more points, so we are going to 'neglect' some of them.

%Crop the calculated contour (we are not fitting the whole contour)

maxexperimental = max(ynorn);
for betai = 1:length(ybeta(1,:))
    maxtheoretical(betai) = max(ybeta(:,betai));
    maxsafecropy(betai) = min(maxexperimental,maxtheoretical(betai));
    for i = 1:length(ybeta(:,betai))
        if(ybeta(i,betai) < maxsafecropy(betai))
            ybetan(i) = ybeta(i,betai);
            xbetan(i) = xbeta(i,betai);
        end
    end

    % For the comparison (calculus of the residues), we only keep to points with the same y
        for j = 1:length(ybetan)
            for i = 1:length(ynorn)
              if((ybetan(j)-ynorn(i))<0.001)
                ydef(j)=ynorn(i);
                xdef(j)=xnorn(i);
            end
        end

        adjustprofilex{betai}=xbetan;
        adjustprofiley{betai}=ybetan;
        adjustprofilex2{betai}=xdef;
        adjustprofiley2{betai}=ydef;
    end
%     figure % take care if you uncomment this figure because it's inside a loop
%     plot(xbetan,ybetan,'o')
%     hold on
%     plot(xdef,ydef,'go-')
%     title(num2str(betai))
%     hold off

    for temp = 1:length(ybetan)
        resids(temp) = xbetan(temp) - xdef(temp);
    end
    stddev(betai) = sqrt(sum(resids.^2)/(length(ydef)-1));
end

bi = 0.05:0.001:0.5;
[kk,betaminind] = min(stddev);
betaval = bi(betaminind);
%
% figure;
% plot(bi,stddev,'o-r')
% title('Chi Squared Values');xlabel('Beta');ylabel('Chi Squared');hold off;


% we have to undo:
% normalize using r
% we deleted one white column of pixels when we tried to find the edge
% we also moved the edge to get (0,0) in the bottom

% figure;
% imshow(inputImgOriginal);hold on;
% plot(xcoordr,ycoordr,'g','LineWidth',2);hold on;plot(xcoordl,ycoordl,'g','LineWidth',2);hold on;
% plot(adjustprofilex{betaminind}*rad+min(xcrop)-1,(-adjustprofiley{betaminind})*rad+max(ycrop)-
1,'bo')
% hold on;title('Original image + Contour + Fitted contour');hold off;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%   INTERFACIAL TENSION FINAL CALCULATION %%%%%%%%%%%%%%%%%%%%%%%%%
deltarho = abs(density1-density2);
preliminarycalc = 9.81*deltarho*(rad/ratio1*(needleTipDiameter/1000)).^2/beta;
secondarycalc = 9.81*deltarho*(rad/ratio1*(needleTipDiameter/1000)).^2/betaval;
set(handles.uipanel1,'Title','Results');
set(handles.uipanel12,'Visible','on');
set(handles.uipanel14,'Visible','on');
set(handles.uipanel12,'Title','Profile Fitting Method');
```

```
set(handles.uipanel14,'Title','DS/DE Aspect Ratio');
set(handles.calc2,'String',num2str(secondarycalc*1000));
set(handles.calc2Nm,'String',num2str(secondarycalc));
set(handles.calc2,'Visible','on');
set(handles.text20,'Visible','on');
set(handles.calc2Nm,'Visible','on');
set(handles.text30,'Visible','on');
set(handles.text23,'Visible','on');
set(handles.origimage,'Position',[565,1,288,352]);
set(handles.text32,'String',num2str(preliminarycalc*1000));
set(handles.text33,'String',num2str(preliminarycalc));
set(handles.text31,'Visible','on');
set(handles.text32,'Visible','on');
set(handles.text33,'Visible','on');
set(handles.text34,'Visible','on');
set(handles.origimage,'Visible','on');
subplot(1,5,[1 2]);imshow(inputImgOriginal);title('Original Image');hold off;zoom on;
subplot(1,5,[3 4]);imshow(inputImgOriginal);title('Contour and Original Image');zoom on;
hold on;plot(xcoordr,ycoordr,'r','LineWidth',2);hold on;
plot(xcoordl,ycoordl,'r','LineWidth',2);hold off;
set(handles.Next1,'Visible','on');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
function getImage_Callback(hObject, eventdata, handles)
global inputImgOriginal
global needleTipDiameter
global density1
global density2
inputImgOriginal = imread(imgetfile);
if ~((density1 == 0) && (density2 == 0) && (needleTipDiameter == 0))
    set(handles.calculate,'Visible','on');
end
set(handles.origimage,'Visible','on');
a = imshow(inputImgOriginal);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
function initialize_gui(fig_handle, handles, isreset)
guidata(handles.figure1, handles);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
function dy =rigid(t,y)
global b123
dy(1,1) = 2 - b123*y(3) - sin(y(1))/y(2);
dy(2,1) = cos(y(1));
dy(3,1) = sin(y(1));
function Next1_Callback(hObject, eventdata, handles)
global rad
global ytemp
global xcoordr
global ycoordr
global xcoordl
global ycoordl
global a0
global mx
subplot(1,5,[1 2]);circle = rsmak('circle',rad, [a0+mx -max(ytemp)+rad]);circ2 = fncmb(circle);
fnplt(circ2);hold on;axis equal;plot(xcoordr,-ycoordr,'r');hold on; plot(xcoordl,-ycoordl,'r');
title(['Radius of the circle: ',num2str(rad)]);zoom on;hold off;
subplot(1,5,[3 4]);circle = rsmak('circle',rad, [a0+mx -max(ytemp)+rad]);
circ2 = fncmb(circle);fnplt(circ2);hold on;axis equal;
plot(xcoordr,-ycoordr,'r');hold on; plot(xcoordl,-ycoordl,'r');
axis([max(xcoordl)*0.6 max(xcoordr)*0.8  -max(ycoordr) -0.95*max(ycoordr)]);
title('Zoom of the bottom of the drop');zoom on;hold off
set(handles.Next1,'Visible','off');
set(handles.Prev1,'Visible','on');
set(handles.Next2,'Visible','on');
function Prev1_Callback(hObject, eventdata, handles)
global inputImgOriginal
global xcoordr
global ycoordr
global xcoordl
global ycoordl
subplot(1,5,[1 2]);imshow(inputImgOriginal);title('Original Image');zoom on;hold off;
subplot(1,5,[3 4]);imshow(inputImgOriginal);title('Contour and Original Image')
hold on;plot(xcoordr,ycoordr,'r','LineWidth',2);hold on;
plot(xcoordl,ycoordl,'r','LineWidth',2);zoom on;hold off;
set(handles.Prev1,'Visible','off');
set(handles.Next2,'Visible','off');
set(handles.Next1,'Visible','on');
function Next2_Callback(hObject, eventdata, handles)
global inputImgOriginal
global xcoordr
```

```
global ycoordr
global xcoordl
global ycoordl
global deind
global deVert
global de
global ds
global bi
global stddev
global betaval
global betaminind
subplot(5,5,[1 2 6 7 11 12 16 17 21 22]);imshow(inputImgOriginal);title('DS DE')
hold on;plot(xcoordr,ycoordr,'r','LineWidth',2);hold on;plot(xcoordl,ycoordl,'r','LineWidth',2);hold
on;
hold on; plot([xcoordl(deind) xcoordr(deind)],[ycoordl(deind) ycoordr(deind)],'go-');hold on;
plot([xcoordl(deVert) xcoordr(deVert)],[deVert deVert],'bo-');
title('Distances DS and DE');text(90,de-90,['DE = ',num2str(de),' pxls'],'BackgroundColor',[1 1 1]);
text(deVert + 15,ds-150,['DS = ',num2str(ds),' pxls'],'BackgroundColor',[1 1 1]);zoom on;hold off;
subplot(5,5,[3 4 8 9 13 14 18 19]);plot(bi,stddev,'*-g');hold
on;plot(betaval,stddev(betaminind),'*r');
title('Beta Chi Squared Minimization');xlabel('Beta');ylabel('Chi Squared Value');
text(0.25,stddev(betaminind)*1.5,['Beta = ',num2str(betaval)]);
zoom on;hold off;
set(handles.Prev1,'Visible','off');
set(handles.Next2,'Visible','off');
set(handles.Next3,'Visible','on');
set(handles.Prev2,'Visible','on');
function Next3_Callback(hObject, eventdata, handles)
global inputImgOriginal
global xcoordr
global ycoordr
global xcoordl
global ycoordl
global adjustprofilex
global betaminind
global rad
global xcrop
global adjustprofiley
global ycrop
subplot(1,2,1);imshow(inputImgOriginal);hold on;
plot(xcoordr,ycoordr,'g','LineWidth',2);hold on;plot(xcoordl,ycoordl,'g','LineWidth',2);hold on;
plot(adjustprofilex{betaminind}*rad+min(xcrop)-1,(-adjustprofiley{betaminind})*rad+max(ycrop)-1,'bo')
hold on;title('Original image + Contour + Fitted contour');zoom on;hold off;
set(handles.Prev2,'Visible','off');
set(handles.Next3,'Visible','off');
set(handles.Prev3,'Visible','on');
set(handles.exit,'Visible','on');
set(handles.rerun,'Visible','on');
function Prev2_Callback(hObject, eventdata, handles)
global rad
global ytemp
global xcoordr
global ycoordr
global xcoordl
global ycoordl
global a0
global mx
subplot(1,5,[1 2]);circle = rsmak('circle',rad, [a0+mx -max(ytemp)+rad]);circ2 = fncmb(circle);
fnplt(circ2);hold on;axis equal;plot(xcoordr,-ycoordr,'r');hold on; plot(xcoordl,-ycoordl,'r');
title(['Radius of the circle: ',num2str(rad)]);zoom on;hold off;
subplot(1,5,[3 4]);circle = rsmak('circle',rad, [a0+mx -max(ytemp)+rad]);
circ2 = fncmb(circle);fnplt(circ2);hold on;axis equal;
plot(xcoordr,-ycoordr,'r');hold on; plot(xcoordl,-ycoordl,'r');
axis([max(xcoordl)*0.6 max(xcoordr)*0.8  -max(ycoordr) -0.95*max(ycoordr)]);
title('Zoom of the bottom of the drop');zoom on;hold off
set(handles.Next3,'Visible','off');
set(handles.Prev2,'Visible','off');
set(handles.Prev1,'Visible','on');
set(handles.Next2,'Visible','on');
function Prev3_Callback(hObject, eventdata, handles)
global inputImgOriginal
global xcoordr
global ycoordr
global xcoordl
global ycoordl
global deind
global deVert
global de
global ds
global bi
global stddev
```

```
global betaminind
global betaval
subplot(5,5,[1 2 6 7 11 12 16 17 21 22]);imshow(inputImgOriginal);title('DS DE')
hold on;plot(xcoordr,ycoordr,'r','LineWidth',2);hold on;plot(xcoordl,ycoordl,'r','LineWidth',2);hold
on;
hold on; plot([xcoordl(deind) xcoordr(deind)],[ycoordl(deind) ycoordr(deind)],'go-');hold on;
plot([xcoordl(deVert) xcoordr(deVert)],[deVert deVert],'bo-');
title('Distances DS and DE');text(90,de-90,['DE = ',num2str(de),' pxls'],'BackgroundColor',[1 1 1]);
text(deVert + 15,ds-150,['DS = ',num2str(ds),' pxls'],'BackgroundColor',[1 1 1]);zoom on;hold off;
subplot(5,5,[3 4 8 9 13 14 18 19]);plot(bi,stddev,'*-g');hold
on;plot(betaval,stddev(betaminind),'*r');
title('Beta Chi Squared Minimization');xlabel('Beta');ylabel('Chi Squared Value');
text(0.25,stddev(betaminind)*1.5,['Beta = ',num2str(betaval)]);
zoom on;hold off;
set(handles.Prev3,'Visible','off');
set(handles.Next3,'Visible','on');
set(handles.Prev2,'Visible','on');
set(handles.exit,'Visible','off');
set(handles.rerun,'Visible','off');
function uipanel1_ButtonDownFcn(hObject, eventdata, handles)
function origimage_ButtonDownFcn(hObject, eventdata, handles)
function exit_Callback(hObject, eventdata, handles)
button = questdlg('Are you sure you want to quit?','Exit Dialog','Yes','No','No');
switch button
    case 'Yes',
        clear all; clear; clc; close all;
        disp('Exiting MATLAB');
        exit;
    case 'No',
        quit cancel;
end
function rerun_Callback(hObject, eventdata, handles)
clear all;clear;clc;close all;
run INTERFACIAL
```

rigid.m

```
function dy =rigid(t,y)
global b123
dy(1,1) = 2 - b123*y(3) - sin(y(1))/y(2);
dy(2,1) = cos(y(1));
dy(3,1) = sin(y(1));
end
```

REFERENCES

[1] Dingle, N. M., Tjiptowidjojo, K. K., Basaran, O. A., & Harris, M. T. (2005). A finite element based algorithm for determining interfacial tension (γ) from pendant drop profiles. Journal Of Colloid And Interface Science, 286(2), 647-660.

[2] Del Río, O. I., & Neumann, A. W. (1997). Axisymmetric drop shape analysis: Computational methods for the measurement of interfacial properties from the shape and dimensions of pendant and sessile drops. Journal Of Colloid And Interface Science, 196(2), 136-147.

[3] Alvarez, N., Walker, L., & Anna, S. (2009). A non-gradient based algorithm for the determination of surface tension from a pendant drop: Application to low Bond number drop shapes. Journal Of Colloid And Interface Science, 333(2), 557-562.

[4] Saad, S. I., Policova, Z. Z., & Neumann, A. W. (2011). Design and accuracy of pendant drop methods for surface tension measurement. Colloids And Surfaces A: Physicochemical And Engineering Aspects, 384(1-3), 442-452.

[5] Peters, F. F., & Arabali, D. D. (2013). Interfacial tension between oil and water measured with a modified contour method. Colloids And Surfaces A: Physicochemical And Engineering Aspects, 4261-5.

[6] F. K. Hansen., & G. Rodsrud. (1991). Journal of Colloid Interface Sci. 141.